BIO LAB 1:

Summary of all the optimization techniques:

i) Genetic Algorithm:

- They are mainly adaptive heuristic / search engine algos that provide solns for search & optimization problems. in ML. It solves unconstrained & constrainted optimization problems based on natural selection.

- Applications:
  → Neural network
  → Data mining
  → image processing

- Adv:
  → optimizes continous, discrete functions
  → Provides solns that can improve over a period.

- Dis Adv:
  → Computationally expensive
  → Time consuming

- Optimization Techniques:
  → Inheritance
  → Mutation

ii) Particle swarm optimization :

- Meta-Heuristic opt. algo. inspired by swarm behavior observed in fishes and birds. It is a simulation of simplified social system.

- Applications :
  → Energy storage optimization
  → Scheduling electrical loads
  → Disease detection & classification

- Adv :
  → Easy scaling of Design variable
  → Derivative free
  → very few algo. parameters.

- Disadv :
  → Slow congruence in search stage.

- Optimization techniques :
  → PSO - CoSN
  → Scaled PSO
  → Disease PSO

iii) Ant Colony optimization:

- Inspired from foraging behaviour of ant colonies. Multi-agent systems are designed based on the social behaviour of bees, termites, ants.

- Application:
  → optimal sol. using travelling salesman
  → Vehicle Routing

- Adv:
  → fast convergence
  → Adaptability

- DisAdv:
  → Locally optimal
  → Slow initial convergence.

iv) Cuckoo search:

- Meta-Heuristic opt. algo. Involves modification of original algo. to improve its effectiveness, such as population reduction.

- Applications:
  → Cloud computing
  → data mining

- Adv:
  → Better tradeoff b/w exploitation & exploration
  → Faster convergence rate.

- Dis Adv:
  → Limited convergence rate with adaptive search

v) Grey Wolf optimization:

- Meta-heuristic Algo inspired by wolves in leadership & hunting. Classifies population into α, β, δ and σ, where α is best soln & σ is worst.

- Applications:
  → Clustering (Neural Network)
  → Power dispatch problem (Robotics & path planning)

- Adv:
  → Simple & easy to implement.
  → Computationally less expensive

- DisAdv:
  → Possible premature convergence
  → Limited scope in search for high dimensional problems.

- Optimization techniques:
  → Differential evolution
  → Particle swarm optimation

vi) Parallel Cellular Algorithm:

- Simple & locally interacting computational units work hand in hand to solve problems. Inspired by biological cellular systems, where each cell updates state based on neighboring cells, typically in a grid like structure.

- Appl:
  → Solving NP-hard problems
  → Network simulation
  → Pattern formations

- Adv:
  → Effective for large scale parallel systems
  → Applicable across various domains

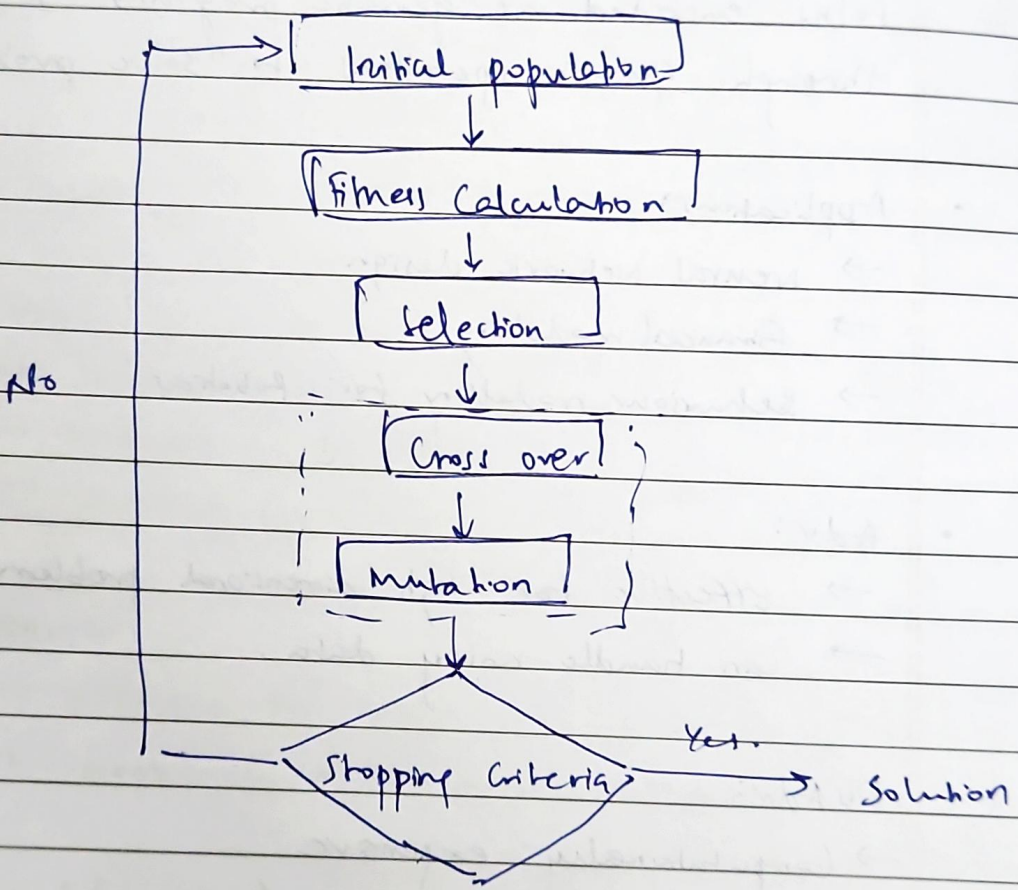
- DisAdv:
  → Difficult to design
  → Limited Global knowledge.

- Optimization techniques:
  → Evolutionary strategies
  → Leader-follower models
  → Hybridization

vii) Gene Optimization Algorithms:

- Evolutionary algos inspired by gene expression. Solns encoded as genetic programs that evolve through genetic operations to solve problems.
- Applications:
  → Neural Network design
  → Financial modelling
  → Behaviour modelling for Robotics
- Adv:
  → Effective for high dimensional problems.
  → Can handle noisy data.
- DisAdv:
  → Computationally expensive
  → Slow convergence rate for complex problems
- Optimization Techniques:
  → Hybridization
  → Incorporating local search after global optimization for refined soln.

BIO LAB 2: GENETIC ALGORITHM:



1) Maximizing function $f(x) = x^2$; $0 \leq x \leq 31$

Step 1: Select Encoding technique

min = 0 .. max = 31 ⇒ (0) 00000 to (31) 11111

Step 2: select intl. pop.

$P_0 = 4$.

String no.	Initial pop. (Random)	x	Fitness $f(x)=x^2$	Prob.	% prob	exp Count	Count
1	01100	12	144	0.1244	12.44	0.4887	1
2	11001	25	625	0.5411	54.11	2.1645	2
3	00101	5	25	0.0216	2.16	0.0866	0
4	10011	19	181	0.3126	31.26	1.2502	1
Sum			1155	1.0	100	4	4
Avg			288.75	0.25	25	1	1
Max			625	0.5411	54.11	2.1645	2

$$\text{Prob} = \frac{F(x)}{\Sigma F(x)} \qquad \text{Expected Count} = \frac{f(x_i)}{\text{Avg } (\Sigma F(x))}$$

String no.	Mating (Random)		Crossover after cross	x	fitness $f(x)=x^2$
1	01100	4	01101	13	169
2	11001		11000	24	576
3	11001	2	11011	27	729
4	10011		10001	17	289
Sum					1763
Avg					440.75
Max					729

String No.	offspring after Cross	Mutation after Flip (Random)	offspring after mutation	x	Fitness $f(x) = x^2$
1	01101	10000	11101	29	841
2	11000	00000	11000	24	576
3	11011	00000	11011	27	729
4	10001	00101	10100	20	400
Sum					2546
Avg					636.5
Max					841

Output:

- Gen 1: 4
  - → Sum: 494
  - → Avg: 123.5
  - → Max: 484
  - → New pop. :→ Sum: 2305
    - → Avg: 461.0
    - → Max: 529

- Gen 2:
  → Sum: 1776
  → Avg: 444.0
  → max: 484
  → New pop.: → Sum: 2860
  → Avg: 452.0
  → max: 484
- Gen 5:
  → Sum: 1776
  → Avg: 444.0
  → max: 484
  → New pop.: → sum: 2836
  → Avg: 567.2
  → max: 900
- Gen 6:
  → Sum: 2910
  → Avg: 740.0
  → max: 900
  → New pop.: → sum: 4129
  → Avg: 825.8
  → max: 900
- Gen 7:
  → sum: 3306
  → Avg: 826.5
  → max: 961
  $x = 31, x^2 = 961$
  Best soln: [30, 31, 29, 37] with $x^2 = 961$

Algorithm:

Step 1) Initialization: Random individual pop. of each represented in binary 5 bits

Step 2) Fitness calc: x^2 calc. for each along with sum, avg & max.

Step 3) Selection: Expected count calc. using $x_i^2 / \text{avg}(x_i^2)$

Step 4) Crossover: Random pairs of individuals Crossover at random bit pos to create offsprings

Step 5) Mutation: Based on mutation rate each individual undergoes mutation flipping 1 bit.

Step 6) Convergence check: Algo. repeats until pop. finds max val. for x^2. here being 961.

BIO LAB 3: ANT COLONY OPTIMIZATION

- Travelling salesman problem.

- Mathematical model to represent :

→ pheromone lvl on graph

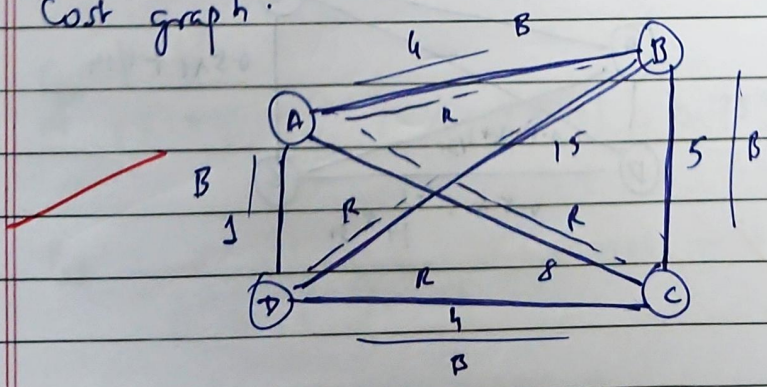
$$\Delta T^{k}_{i,j} = \begin{cases} 1/L_k & k^{th} \text{ ant travels on edge } i,j \\ 0 & \text{otherwise} \end{cases}$$

(L_k : length of path)

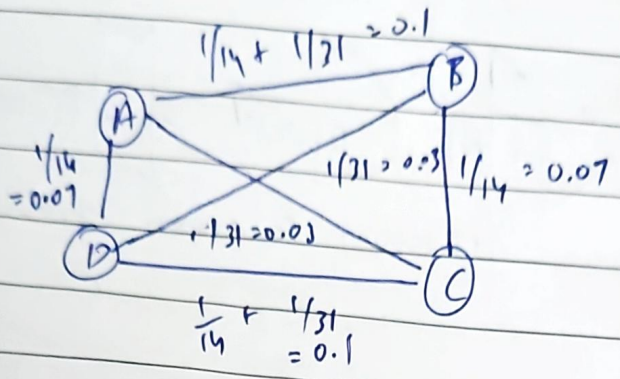
$$e^{k}_{i,j} = \sum_{K=1}^{m} \Delta T^{k}_{i,j} \rightarrow \text{with vaporization}$$

$$T^{k}_{i,j} = (1-\rho)\, T_{i,j} + \sum_{k=1}^{m} \Delta T^{k}_{i,j} \quad \text{with vaporization}$$

- Cost graph :



Red ant - R
Black ant - B



➢ No evaporation:

$R \rightarrow L_1 = 15 + 4 + 4 + 8 = 31$

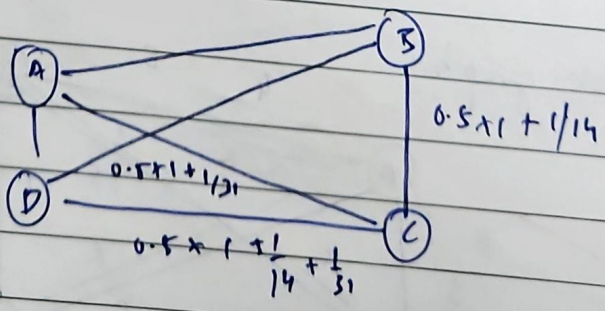
$B \rightarrow L_2 = 1 + 4 + 5 + 4 = 14$

$\Delta r^1_{ij} = 1/14$ } Pheromone lvl on path
$\Delta r^2_{ij} = 1/31$ } by ant.

➢ With evaporation:

$S = 0.5$

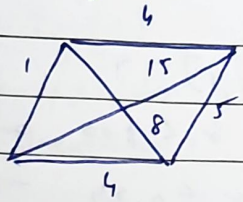
$c_{ij} = 1$



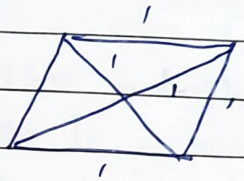
Probability of choosing path i,j

$$P_{i,j} = \frac{(c_{i,j})^{\alpha} (n_{i,j})^{\beta}}{\Sigma ((c_{i,j})^{\alpha} (n_{i,j})^{\beta})}$$

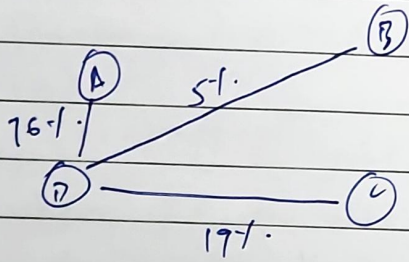
$$n_{i,j} = \frac{1}{L_{i,j}}$$



Cost graph



Initial pheromone lvls



$$P_{DA} = \frac{1 \times 1/1}{(1 \times 1/1) + (1 \times 1/15) + (1 \times 1/4)} = 0.7595 = 75.95\% \sim 7$$

$$P_{DB} = \frac{1 \times 1/15}{(1 \times \frac{1}{1}) + (1 \times \frac{1}{15}) + 1 \times \frac{1}{4}} = 0.0506 \sim 5\%$$

$$P_{DC} = \frac{1 \times 1/4}{(1 \times \frac{1}{1}) + (1 \times \frac{1}{15}) + (1 \times \frac{1}{4})} = 0.1899 \sim 19\%$$

- Roulette wheel : to choose destination

	A	C	B
Probabilistic	0.76	0.19	0.05

	A	C	B
Cumulative sum	0.76 + 0.19 + 0.05 = 1	0.19 + 0.05 = 0.24	0.05

random no. (r) in [0,1]

if $0.24 < r < 1.00 \rightarrow A$

if $0.05 < r \leq 0.24 \rightarrow C$

if $0.00 \leq r \leq 0.05 \rightarrow B$

- Algorithm

```
TSP-ACO()
    best tour ← nil
    repeat
        randomly place M ants on N cities
        for each ant k
            for n ← 1 to N
                ant k select edge from P_n^q
        update best tour
        for each ant q
            for each edge (q, v) in ant j tour
                deposit pheromone α 1/tour length of (u,v)
    until termination criteria.
    return best tour.
```

on city i the k^{th} ant moves to city j with probability:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} + [\eta_{ij}]^{\beta}}{\sum_{l \in \text{allowed}_k} ([\tau_{il}(t)]^{\alpha} + [\eta_{il}]^{\beta})} & \text{if } j \in (t) \\ 0 & \text{otherwise} \end{cases}$$

where ant k allowed to

→ where, $\tau_{ij}(t)$ is pheromone on edge ij is called visibility 1/d to distance. b/w i & j.

→ Updating pheromone: Each ant deposits Q/Lk pheromones on edges traversed

Q → constant L_k → cost of tour

→ Total pheromone deposited on edge ij is $\Delta\tau_{ij}$ $(t, t+n)$

$$\tau_{ij}(t+n) = (1-\rho)^{3}\tau_{ij}(t) + \Delta\tau_{ij}(t, t+n)$$

where ρ is the rate of evaporation of pheromone.

#2
23/1/20
Sec

BIO LAB 4: PARTICLE SWARM OPTIMIZATION

PSO, or particle swarm optimization, is a meta-heuristic optimization algorithm inspired by social behaviour of bird flocks.

This algo. works by initializing a population of particles, each representing a solution to the problem.

In each iteration, the particles update their position & velocity based on the following rules:

1. Personal / Individual best:
   Each particle keeps a track of its best position (pbest)
2. Global / swarm best:
   The best position found by any particle in the population is called swarm best (sbest)

3. Previous velocity:
Represents particle's momentum from previous iteration, influencing its current direction of moment.

Particle position:

$$x_{k+1}^i = x_k^i + v_k^i$$

Particle velocity:

$$v_{k+1}^i = w_k v_k^i + c_1 r_1 (p_k^i - x_k^i) + c_2 r_2 (p_k^g - x_k^i)$$

where x_k^i = particle position
v_k^i = particle velocity
p_k^i = best individual particle pos.
p_k^g = best swarm position
w_k = constant inertia weight
c_1, c_2 = cognitive & social parameter
r_1, r_2 = random no.s 0 & 1

For particle velocity eqn
i) Social term: $c_2 r_2 (p_k^g - x_k^i)$
ii) Cognitive term: $c_1 r_1 (p_k^i - x_k^i)$

Algorithm:

Step 1: Initialize the swarm > Randomly generate a population of particles & assign random initial position & velocities to each.

Step 2: Inhalize the parameters w (weight) cognitive & social coeff. C_1 & C_2 respectively.

Step 3: Evaluate fitness value of each particles current position using objective function.

Step 4: Identify particle with best fitness value & set global best pos. to particle with best fitness.

Step 5: Update velocities and position with the equations:-

$$v^i_{k+1} = w_k * v^i_k + c_1 r_1 (p^i_{ic} - x^i_k) + c_2 r_2 (p^g_k - x^i_k)$$

&

$$x^i_{k+1} = x^i_k + v^i_{k+1}$$ respectively

Step 6: Terminate if max iterations have reached otherwise back to step 2.

~~Step 7 black~~

O/P
30/10

Bio LAB 5: CUCKOO SEARCH ALGORITHM

CSA is a nature-inspired optimization algorithm based on brood parasitic behaviour of cuckoos.

Used in solving complex optimization problems where traditional methods like gradient-descent or other methods may not be applicable.

~~Aim~~

Purpose:

The Purpose of CSA is to:

i) find optimal solution: Helps find optimal soln where sol. space is vast and complicated.

ii) Exploit Randomization: Algo exploits ~~random~~ randomness in nest ~~pla~~ ~~randomness~~ placement and search process.

iii) Efficient performance: CSA is known for its simplicity & effectiveness and ability to balance exploitation & exploration.

Applications:

- Structural design optimization
- Image segmentation & compression
- Training machine learning models by optimizing loss functions.
- DNA sequencing.

Algorithm:

The CSA is inspired by the following principles:

i) Cuckoo Bird's laying eggs:

Female Cuckoo lays eggs in nest of other birds (host)

ii) Host nest selection:

Host bird may detect foreign eggs & throw them out.

iii) Nest's evolution:

Nests evolve based on quality of their eggs (solutions).

The algo combines 2 strategies:

- Levy flight: Random walk process that ensures large movements in search space to explore new areas. (exploration)
- Local search: Refining solutions by exploiting areas around the current solution. (exploitation)

Mathematical formulation / Mathematical Algo:

→ Initialization: random pop. n candidate solutions (nests) in search space.

→ Evaluate fitness of candidates using objective function.

→ Levy flight update: Each cuckoo updates its position using this distribution

$$x_i(t+1) = x_i(t) + \alpha \cdot \text{Levy}(\lambda)$$

~~where $x_i = n$~~

where $x_i(t+1)$ is the new position of i^{th} nest.

$x_i(t)$ is current position

α is step size

Levy (1) is levy distribution used for generating random steps

→ Greedy replacement rule:

After levy flight update, algo checks if new pos. better than old one. if new pos. better (lower) obj. val., it replaces old nest. (Exploitation part of algo.

$$x_i^{new} = \begin{cases} x_i^{new} & \text{if } f(x_i^{new}) \le f(x_i) \\ x_i & \text{otherwise} \end{cases}$$

$f(x_i^{new})$ is val of new nest after applying levy flight.

13/11

→ Host nest Abandonment!

Host may discover foreign egg (intersn sol.) & throw it.
prob p_a where $0 \leq p_a \leq 1$ showing likeliness it nest will be abandoned

$$x_i^{new} = \begin{cases} x_i^{new} & \text{with prob. } p_a \\ x_i & \text{with prob } (1-p_a) \end{cases}$$

Here p_a is probability of abandoning current sol.

BIO LAB 6:

GREY WOLF OPTIMIZATION

Purpose:

Nature inspired optimization algo based on social heirrarcy & hunting behaviours of grey wolves in the wild. The primary purpose is to find optimal sols to complex multi-dim & non-linear opt. problem.

Applications:

1. AI & ML: Hyperparameter of machine learning algo, calc & neutral n/w opt.
2. Img. processing: Img segmentation, edge detection & other tasks
3. Robotics: Path-planning, multi-robot coordination & task scheduling.

Algorithm:

i) Init: Grey wolves pop. is init. randomly within search space.

ii) Social heirarcy:

4 categories:

- Alpha (α): Leader
- Beta (β): second best soln.
- Delta (δ): better than others but less than α & β.
- Omega (ω): remaining wolves, random solution.

iii) Hunting mechanism:

$$D = |C \cdot X_p - X_i|$$

D → dist. b/w prey & wolf.
C → coeff. that controls dist.
X_p → pos. of grey.
X_i → pos. of wolves.

iv) pos. update:

$$X_i(t+1) = X_i(t) + A \cdot D$$

A → coeff that changes over time to guide search process.

Implementation:

1. Init the population.
2. Evaluate fitness
3. Identify best wolves based on dist btw prey & wolves
4. Update pos.
5. Iterate steps from 2-4 till convergence criteria is met.
6. End.

~~Output:~~

20/11 op

Best pos. = [-1.393, 1.7221, -4.537, -1.936, 1.477]

Best score: 2.857.

Bio Lab 7: Parallel Cellular Algorithm.

Purpose:

PCA is a computational approach used to solve complex problems through parallel processing of data. It is inspired by cellular automata (CA), where the problem is divided into small local regions or cells, that operate in parallel, updating states based on predefined rules.

Applications:

1. Image processing: Helps in image filtering, segmentation and edge detection.

2. Computational biology: PCA helps in simulating biological processes or analysing large scale genome data.

Algorithm:

i) Initialization:

Grid representation: The grid is intl. as a 2D array of cells.
(C)

Neighbouring cells: In simplest case, each cell has 8 neighbours defined as -

$$N(x,y) = \{(x-1, y-1), (x-1, y), (x-1, y+1), (x, y+1), (x+1, y-1), (x+1, y), (x+1, y+1)\}$$

Also check the fitness of each cell & $F(x,y) = F(C(x,y), N(x,y))$

ii) Update rule:

for each cell $C(x,y)$, the next state $C'(x,y)$ depends on current state & neighbours

General update rule:

$$C'(x,y) = F(C(x,y), N(x,y))$$

iii) Parallelization:

The computation of each cell depends on its neighbours, making it suitable for parallel processing. For each time step, update is computed simultaneously for all cells.

The time grid t is rep. as C_t & state of time grid at time t+1 is:

$$C_{t+1} = \{C'(x,y) : (x,y) \in Grid, F(C, (x,y), N(x,y))\}$$

This op. is performed in // for all cells in grid.

iv) For termination condition:

The termination happens when grid reaches stable state, or a predefined no. of iterations.

$\| C_t - C_{t+1} \| F = 0$. (no change b/w successive grids)

Implementation:

1. Initialization:

Initialize grid ((x, y) for time t=0 along with the fitness function for each cell.

2. Update parallel state:

For each time step, use parallel approach to update all cells. Parallel nature ensures all cells are updated at once, reducing computation time significantly.

3. Synchronization

4. Repeat / Terminate.

Output:

For GRID 6:

Init!	Iteration 1	Iteration 2.
#..###	.#.#..	.#.....
....#.	..#...	.....
.#.#.#	.###..	...#..
##..##	.##...	.#.#...
#....#	.....	.##...
##..#	.#....	..#...

Bio LAB 8: OPTIMIZATION VIA GENE EXPRESSION

Purpose:

To optimize for complex problems by simulating the process of gene expression, leveraging the principle of natural selection & evolutionary computation.

Application:

Resource allocation
Scheduling problems
path optimization

Algorithm :

- Initialization: Generate a pop. of candidate sol^ (chromosomes) rep by binary string.

- Gene exp to phenotype conversion use a function $f(g)$ to map the genotype to a phenotype

- Fitness Evaluation: Evaluate fitness $F(p_i)$ of each using an objective function $F(p)$
  ex: $F(p) = -p^2 + 5p$

- Selection: Select parents based on their fitness using roulette wheel selection

$$P(g_i) = \frac{F(p_i)}{\sum_{i=1}^{n} F(p_i)}$$

- Cross over: Combine two parents genes g_A & g_B to create offspring

$$g_{child} = \alpha \cdot g_A + (1-\alpha) \cdot g_B$$

- Mutation: Mutate a gene g_i by adding a small random value δ:

$$g_i = g_i + \delta_i$$

- Repeat steps 2 to 6 until the prog. converges max no of generation is matched.

Output:

Best sol: 8.417.
Fitness: 7.085